Critical Infrastructure Protection A Real Time Alerting System: Tools & Models

Secure Mediation

Alessandro NERI

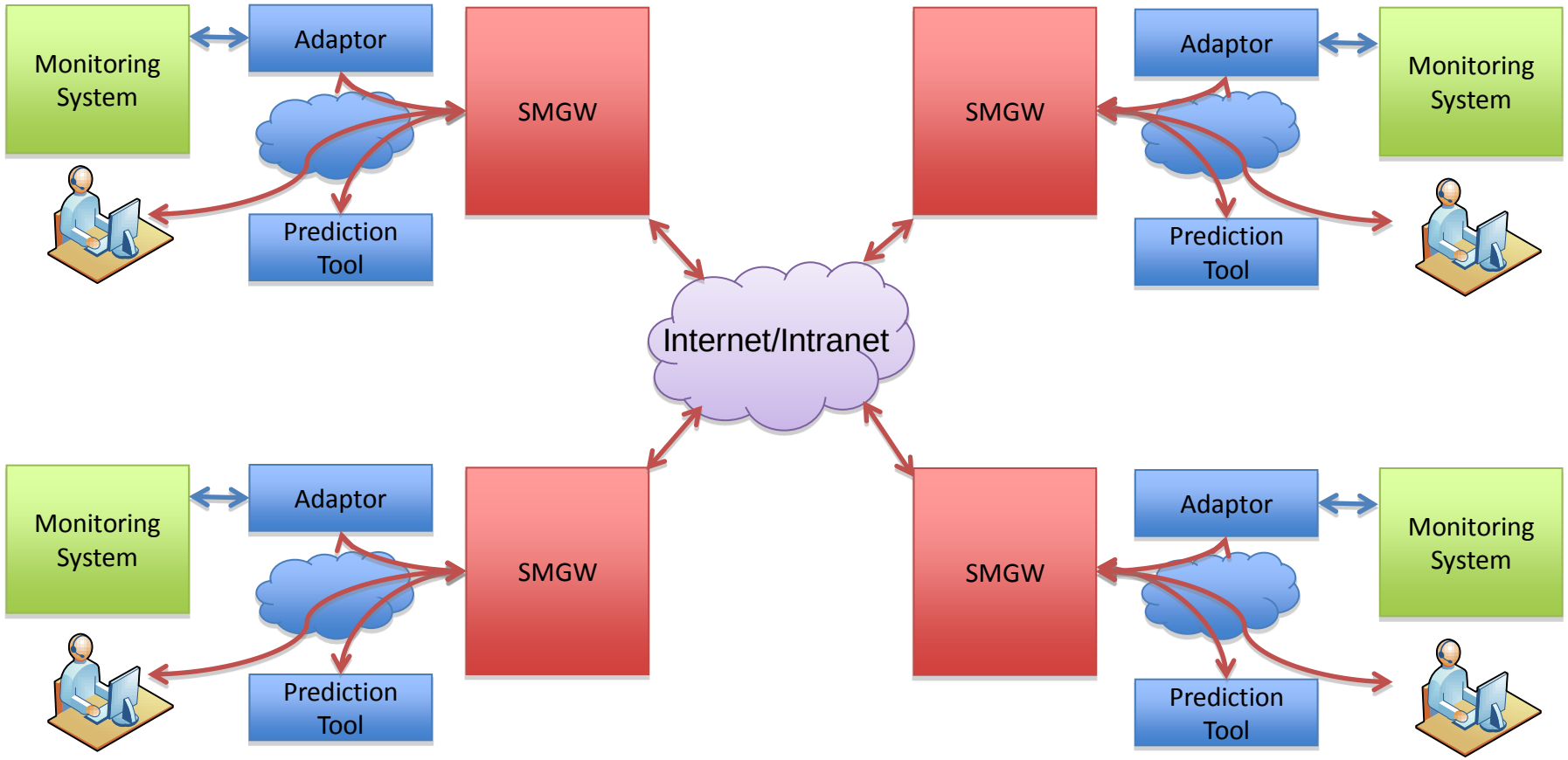
ROMA – February 28th 2011

www.micie.eu
ICT-SEC 225353

Contents

- Secure Mediation requirements
- Secure Mediator roles
- Security model
- Logical Architecture
- Deployment architecture & technology stack
- Prototype preview

MICIE General architectural model



SMGW

The SMGW is a system implemented in each CI allowing reliable and secure communication and data exchange with other CIs.

Each SMGW interfaces with the following main entities:

–The Adaptor of the local Critical Infrastructure through the SMGW Adaptor Standard i/f;

–The local Prediction Tool through the SMGW-PT i/f;

–Remote SMGWs clients of other CIs through the SMGW Web Services Application Server;

–Remote SMGW Web Services Application Servers of other CIs through the SMGW Clients;

–Users interacting with the SMGW applications (e.g., Administration, Operation & Maintenance, Subscription &Security Policy Management, Test, Simulation) through browser based Human Machine Interfaces (HMIs).

- Security objectives identification (1/3)
- **Data availability**:
  - R4.2.1 (HR): O.Data_Exch_Rel_Ava – The TOE shall provide availability and reliability between SMGWs
  - R.4.2.3 (R): O.Mediat – The TOE shall mediate the flow of all information from users and shall ensure that residual information from a previous information flow is not retransmitted
- **Data integrity**:
  - R4.2.4 (HR): O.Data_Exch_Int – The TOE shall provide control and protection of integrity between SMGWs, on the basis of cryptographic rules of exchange
  - R4.2.5 (HR): O.Crypto_Key – The TOE shall provide protection of cryptographic keys during their storing, functioning and destruction
  - R4.2.6 (R): O.Data_TimT – The TOE shall provide temporal identification of data by timestamping

- Security objectives identification (2/3)
- **Data confidentiality**:
  - R4.2.11 (HR): O.Data_Exch_Conf – The TOE shall provide control and protection of confidentiality between SMGWs, on the basis of cryptographic rules of exchange
  - R4.2.15 (O): O.Policy_Admin – The TOE shall be managed according to a specific policy, including the filtering and cryptographic rules used
- **Traceability**:
  - R4.2.16 (R): O.Account – The TOE shall provide user accountability for information flows through the TOE and for authorized administrator use of security functions related to audit
  - R4.2.17 (R): O.Audrec – The TOE must provide a means to record an audit trail of security-related events

- Security objectives identification (3/3)
- **Non-repudiation**:
  - R4.2.21 (R): O.Data_Filter – The TOE shall provide data exchanges filtering between SMGWs, on the basis of data filtering rules
- **Secure environment** (to protect the environment where the TOE is installed):
  - R4.2.28 (HR): OE.Terminal – The terminal used to manage the dedicated interface shall be protected against from any device allowing to capture secret elements of the configuration of the TOE during its local administration
  - R4.2.33 (O): OE.D_Int_Protct – The TOE shall be appropriately physically protected; individuals responsible of the dedicated interface shall install it in secured environment able to prevent non-authorized physical access

MSGW Roles

- Acquires information related to the local CI through a CI-specific adaptor
- Discovers information related to remote CIs by means of a secure communication with remote SMGWs
- Stores all the information in a dedicated DB
- Interacts with the prediction tool for the provisioning of the required information
- Delivers current and predicted CI status to subscribers (HMI, other MSGWs)
- Manages the security policies
- Stores client identities in terms of client profiles, keys and certificates.
- Authenticates and authorizes any client access (federation model).
- Applies security mechanisms to SOAP and plain XML messages (encryption/decryption and signature/authentication of different parts of an XML message)
- Performs security auditing

***IntraCI mode**:* refers to the communications between the CI adaptor and the adaptor interface that constitutes a part of the SMGW as well as to the communications between an eventually remote prediction tool and its SMGW interface.

IntraCI mode

- Based on IPSec
- Protection of local calls to the SMGW

InterCI mode

- Transport level
  - SSL/TLS (*HTTPS*)
- Application-level (*Web Services*)
  - Confidentiality, Integrity, Authenticity
    - XML Encryption, XML Signature
  - Message Structure, Message Security
    - WS-Security
  - Metadata
    - WS-Policy

***InterCI mode**:* refers to the communications with other SMGWs, including those belonging to the same CI owner, as well as to the communications with remote HMI clients and analysis tools, including those running on hosts directly connected to the same LAN to which the SMGW is connected.

IntraCI mode

- Based on IPSec
- Protection of local calls to the SMGW

InterCI mode

- Transport level
  - SSL/TLS (*HTTPS*)
- Application-level (*Web Services*)
  - Confidentiality, Integrity, Authenticity
    - XML Encryption, XML Signature
  - Message Structure, Message Security
    - WS-Security
  - Metadata
    - WS-Policy

IntraCI mode

•Considering that only a limited number of adaptors (typically 1) will be interfaced to the same SMGW and that, in this case, the link capacity and protocol overhead are of primary concern, the *IntraCI* mode will adopt the ***IPSec protocol*** to secure the communications with adaptors and prediction tool.

•In addition, metadata provided by the adaptor shall by signed by the adaptor itself and time stamped using the private key associated to the adaptor ***X.509 certificate***.

•The adaptor interface will authenticate the received metadata before storing them in the SMGW database.

InterCI

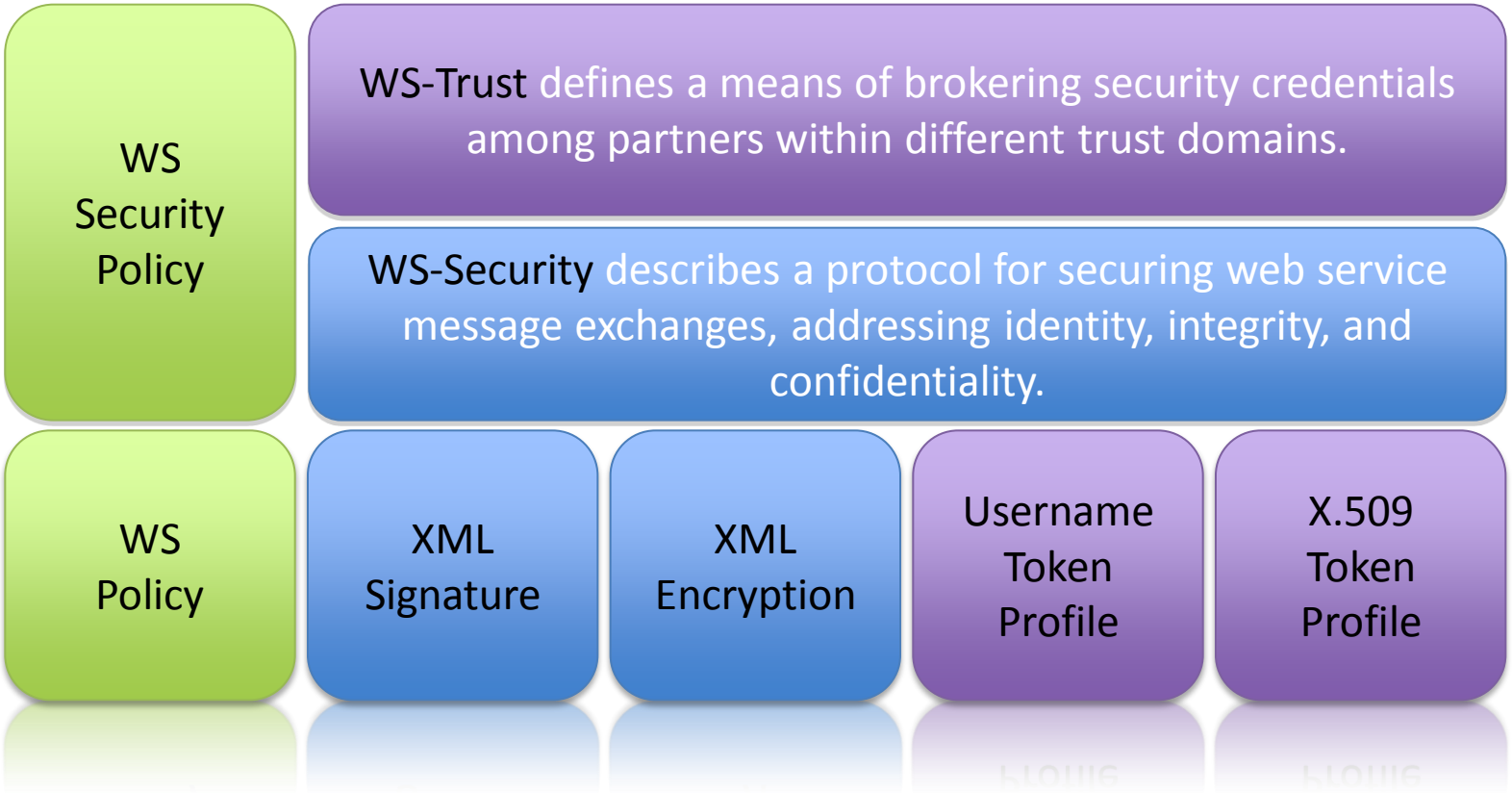
•In order to efficiently and effectively support both request/response and publish/subscribe mechanisms to access information services provided by the SMGW from a huge number of nodes distributed over Internet, a **Web Services oriented architecture** is adopted at application level.

•Consequently, the *InterCi* mode will employ the HTTPS protocol to secure the connections between clients and servers.

•Although less efficient than IPSec with respect to latency, computational complexity, and overhead, this solution presents a simpler security policy management, especially when the federation model is considered.

•However, since simply controlling access to web services does not fulfill all the SMGW security requirements, a finer granularity of the authorization, authentication and encryption mechanism at application level is adopted by the *InterCI* mode

The WS Security Platform

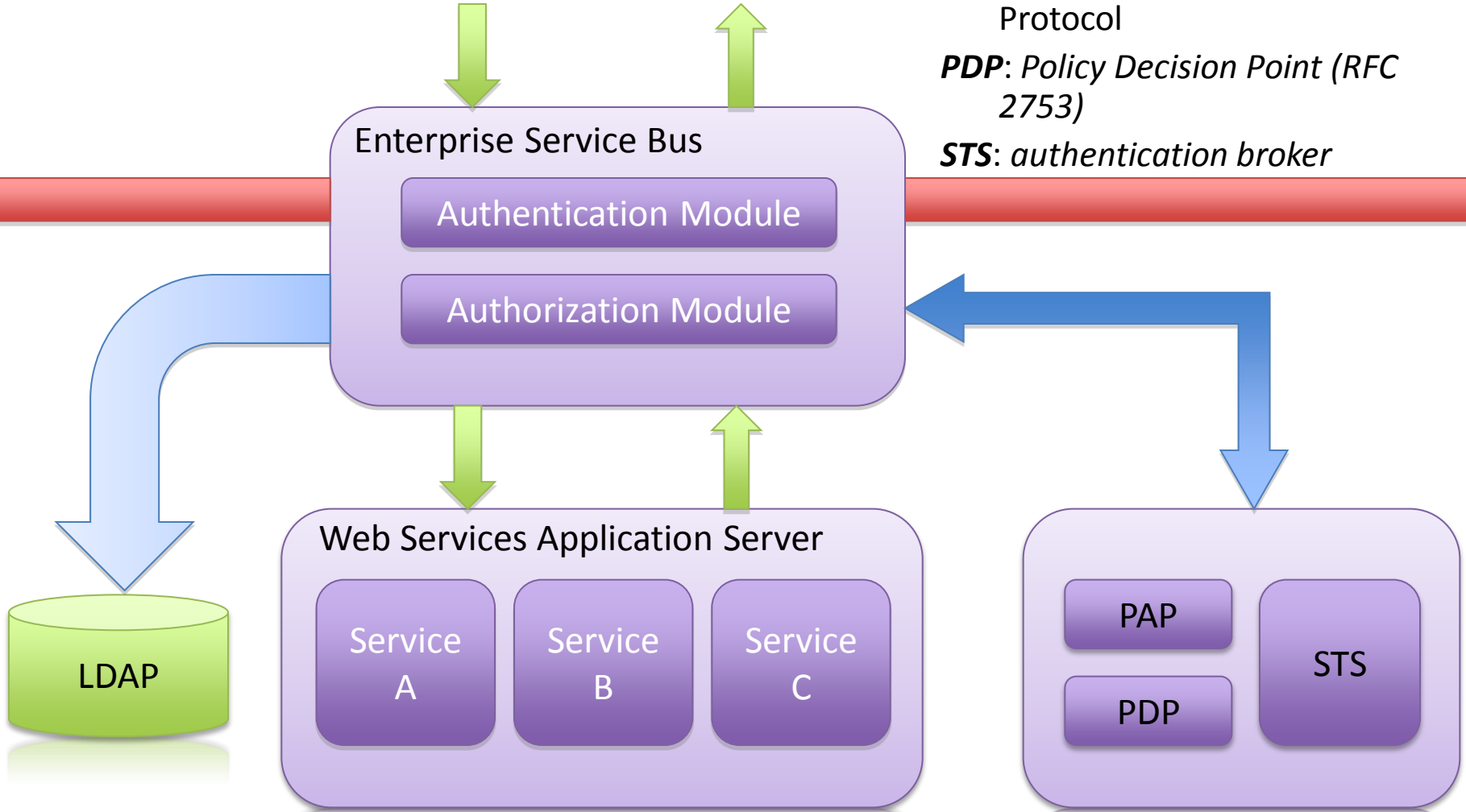


WS Security Reference Architecture

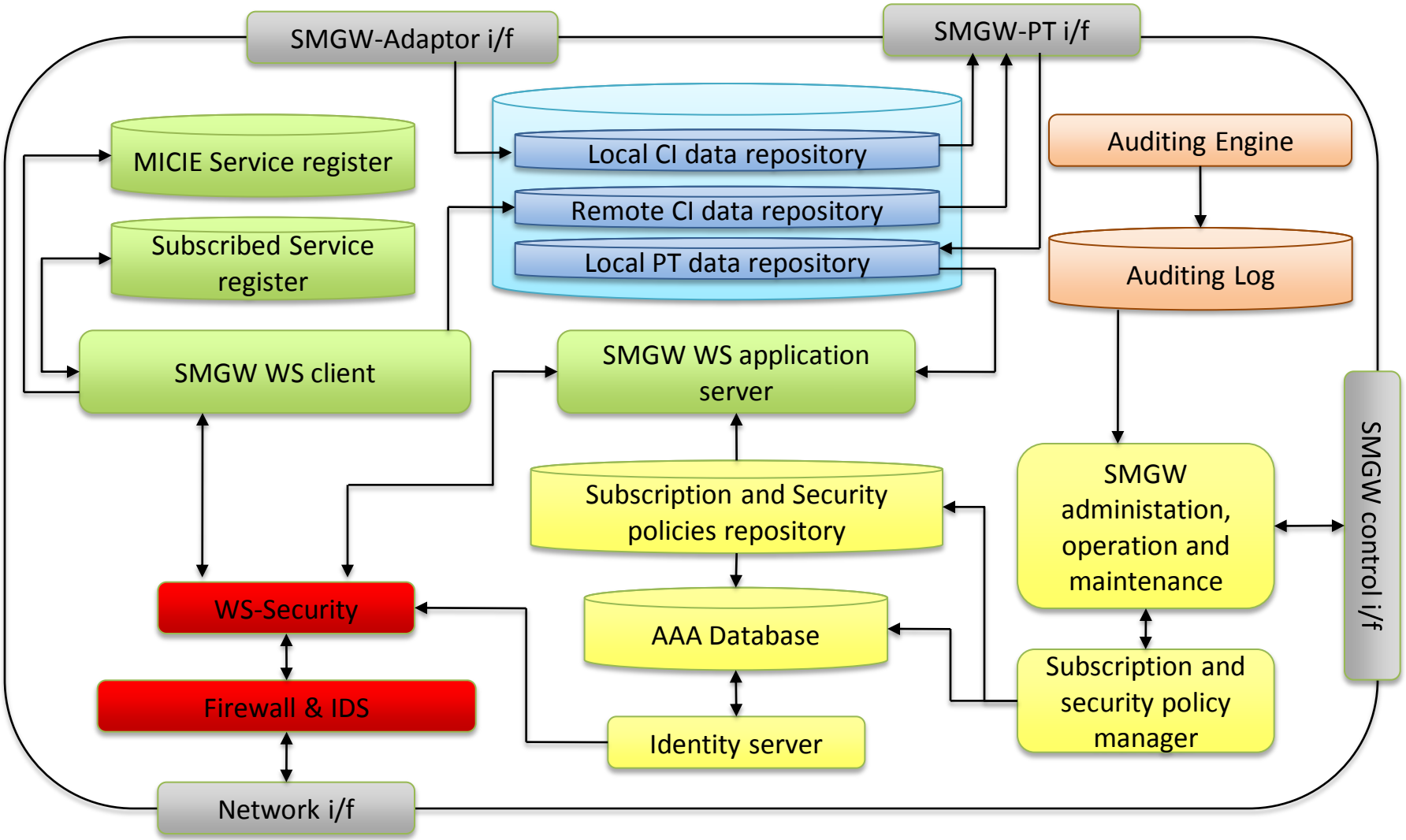
PAP: Password Authentication Protocol

PDP: *Policy Decision Point (RFC 2753)*

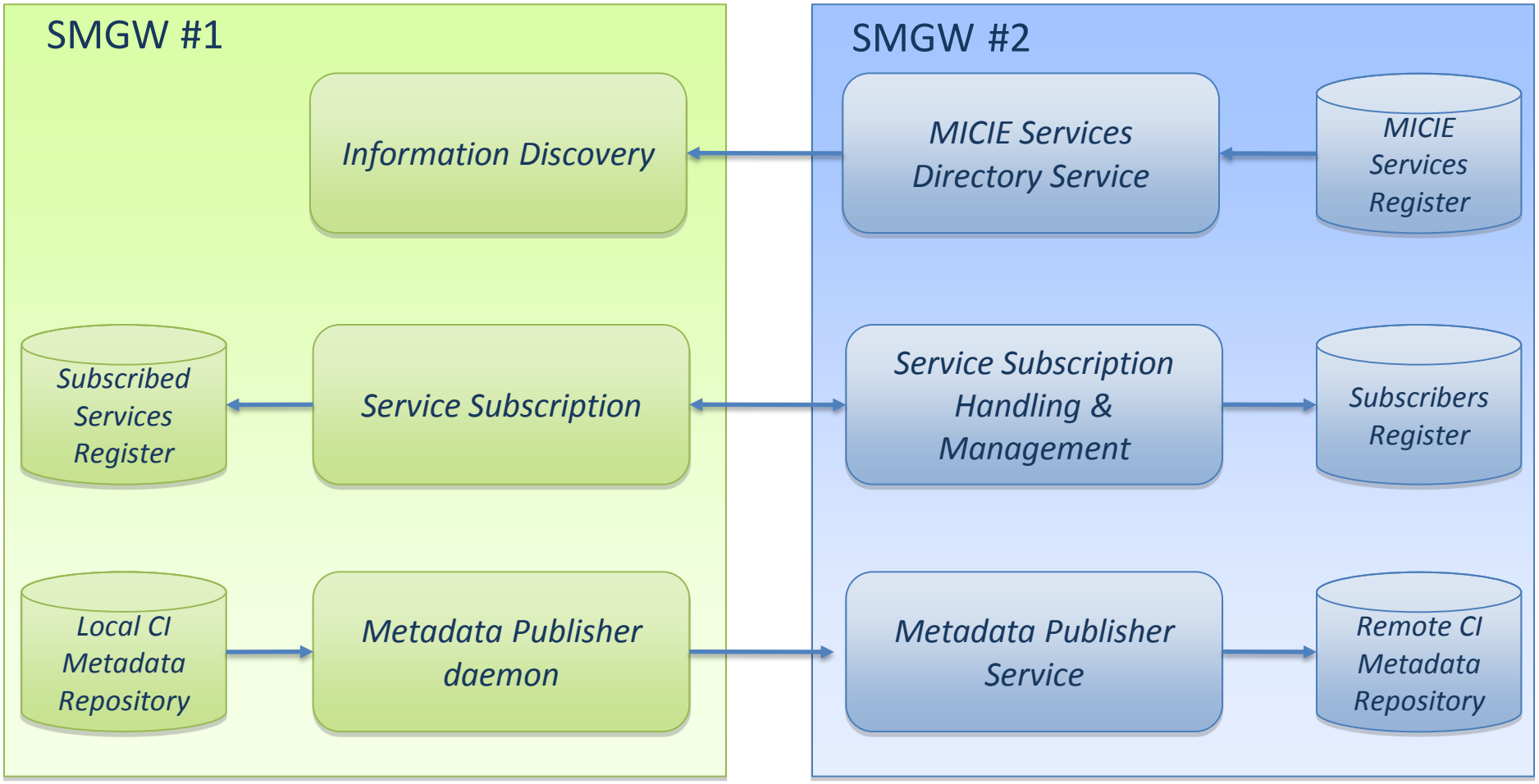
STS: *authentication broker*



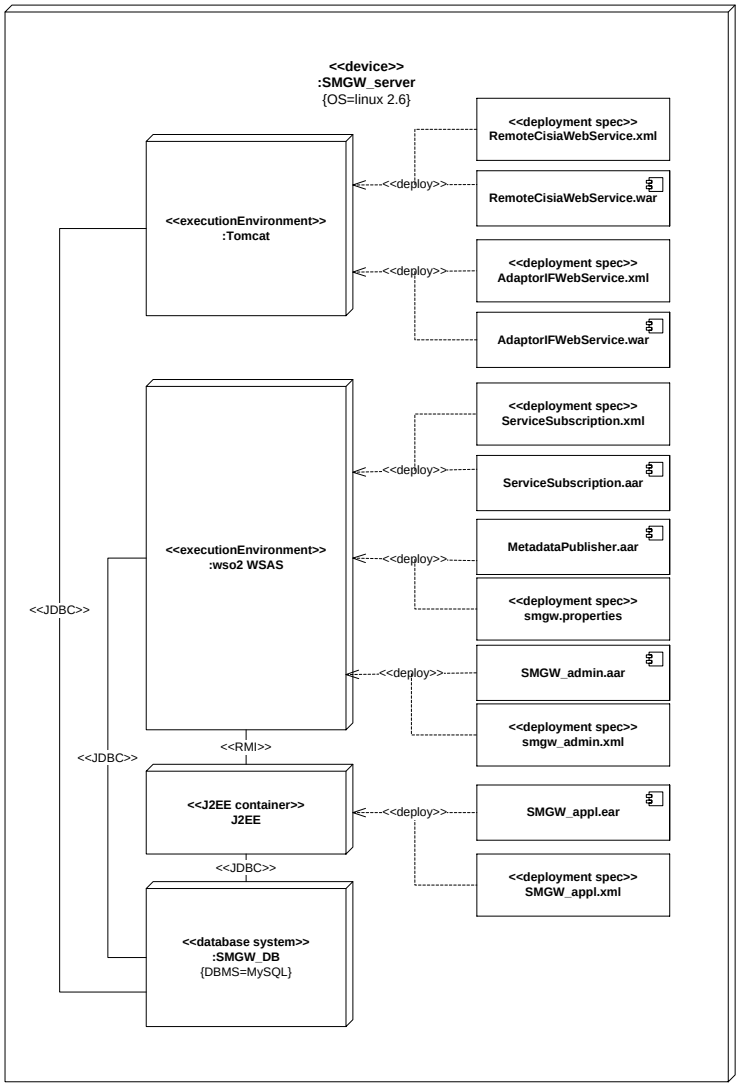
SMGW Logical Architecture



SMGW – Inter CI interactions

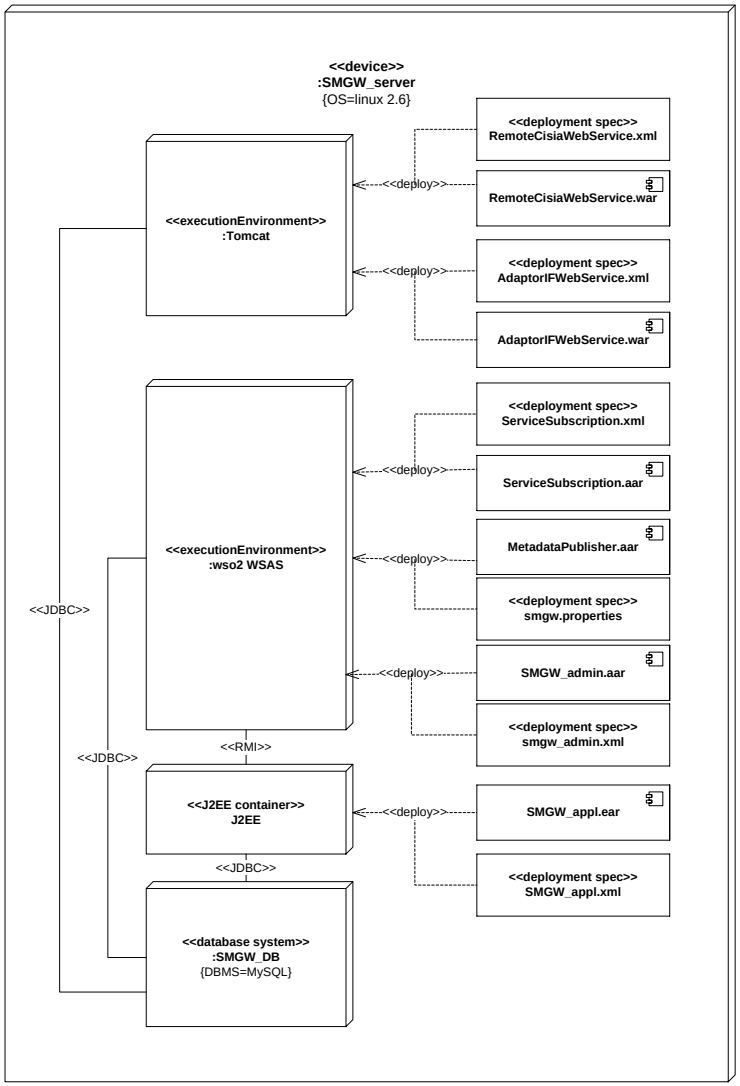


- **Multiple platforms supported:**
  - **Demonstrator: Linux** operating system
  - tested also on Mac OSX
  - The solution should also run on Microsoft Windows Computers
- **Apache** foundation open source technological stack for Web Services
- Open source **wso2 oxygen** middleware providing a browser based interface for the setup of the configuration files related to Apache stack
- **Eclipse** open source IDE platform
- Limited number of artifacts

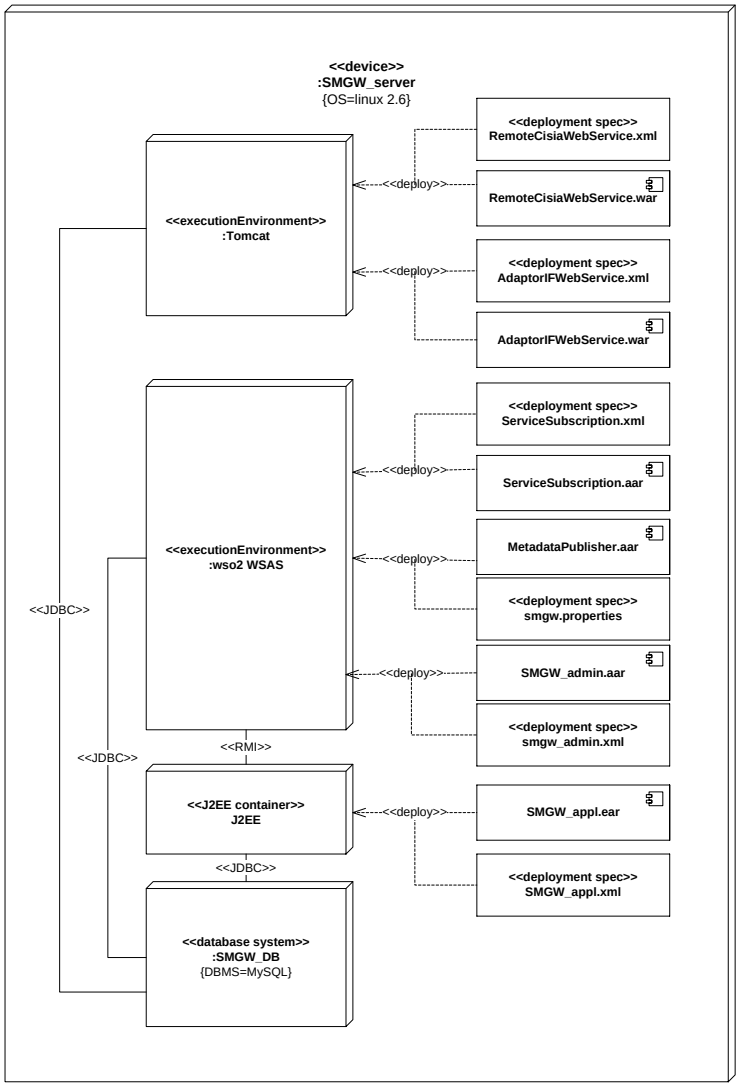


- ***ServiceSubscriptionRequest*** *Web Service*: handles the requests presented by the ServiceSubscriber artifact of other SMGWs.
- The information about requests and acceptance notifications are directly stored/retrieved from the local DB.
- At this aim, it shall provide at least the following operations:
  - ***submitSubscriptionRequest***
  - ***updateSubscriptionRequest***
  - ***getSubscriptionStatus***
  - ***unsubscribeService***

Deployement architecture

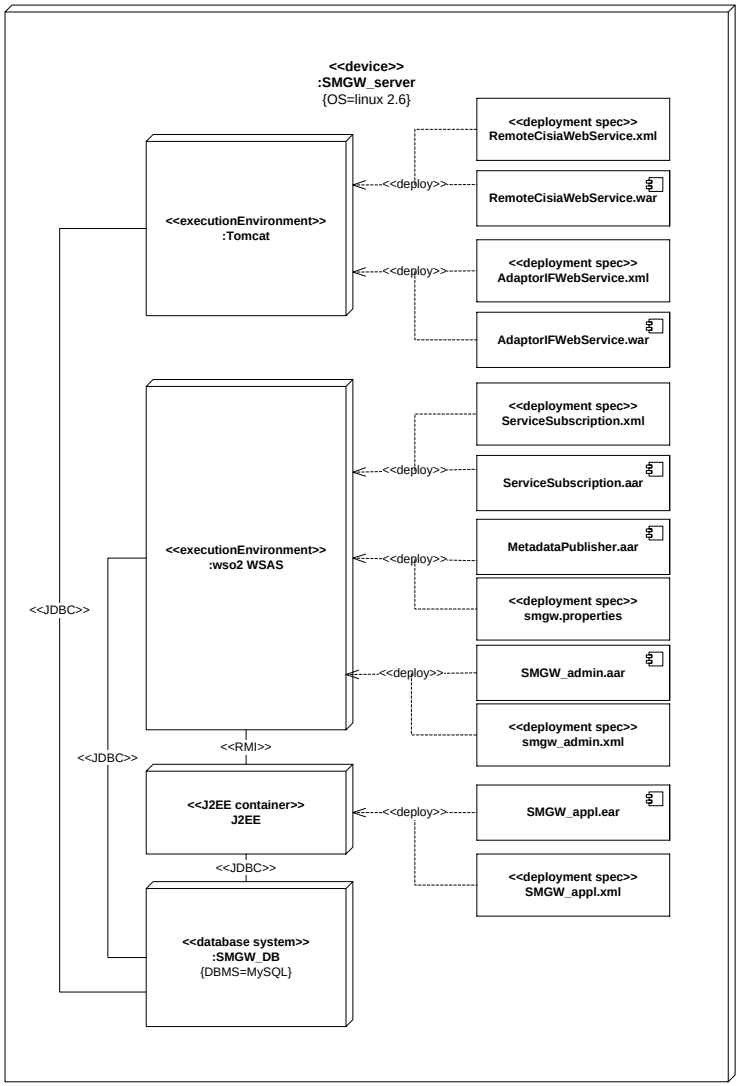


- ***MetadataPublisher Web Service***. It is a Web Service that implements the serverside *Information sharing framework* through an Apache Axis 2 web service.
- It provides the following operations:
  - ***publishMetadata***, that receives the updated datasets from remote SMGWs and stores them in the Remote CI Data Repository (push mode).
  - ***getPrediction***, that retrieves and delivers the available datasets from the CI local Metadata Repository.
  - ***listServices***, that retrieves and delivers the information on the available services.

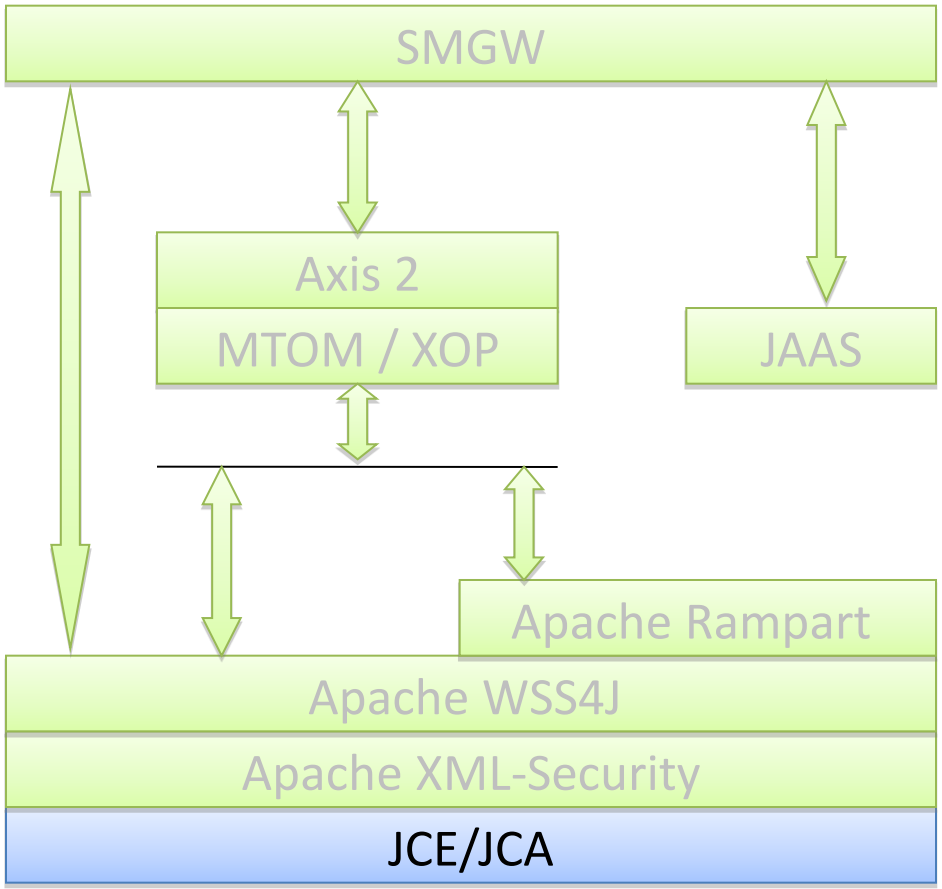


- ***SMGW_admin***. Functionalities related to management, operation and maintenance of the SMGW are supported by :
  - the wso2 oxygen console
  - a set of Java Server Pages, providing all those functions specific of the SWGW (e.g. subscription handling).
- The *SMGW_admin* supports at least the following operations:
  - ***listPendingRequests***
  - ***listSubscriptions***
  - ***updateSubscriptionStatus***
  - ***getSubscriptionStatus***

Deployement architecture

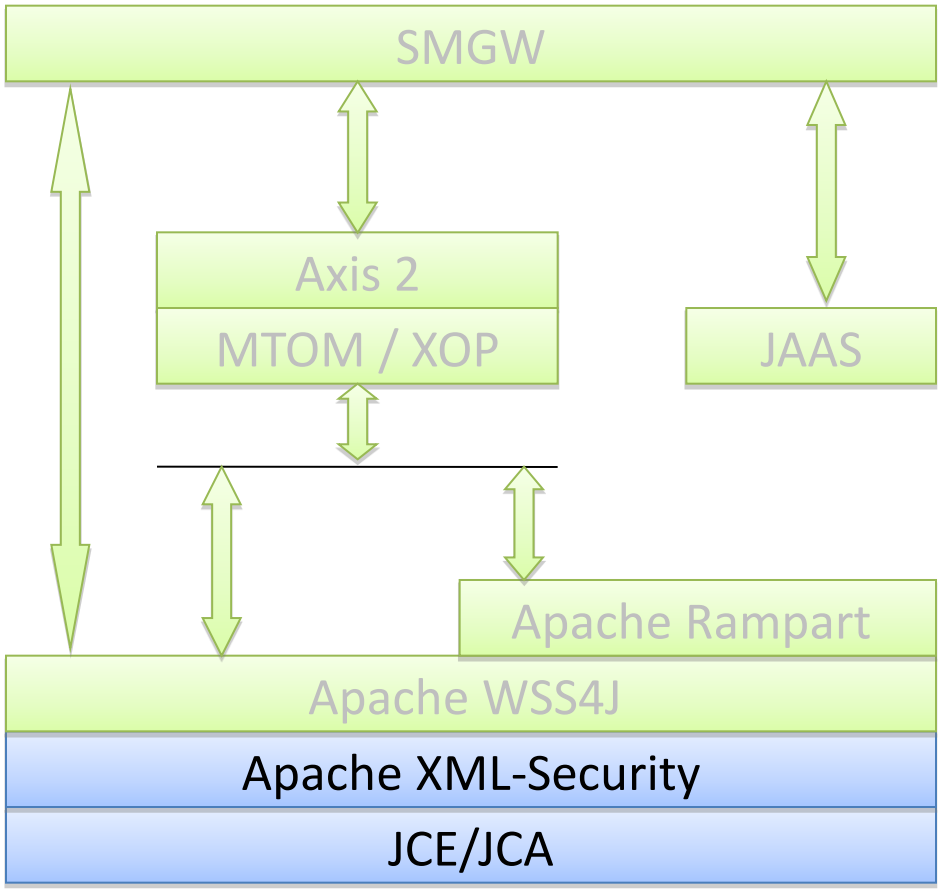


- ***SMGWdaemon*** implements:
  - SMGW Operational Sequences and WorkFlow Control
  - Event Notifications Handler
  - Clientside Information Discovery & Metadata Retrieval
  - Metadata Publisher (clientside push mode),
  - Prediction Tool Interface w.r.t. to incoming data (PT provider pull mode),
  - Adaptor Interface w.r.t. to incoming data (Adaptor pull mode)
  - Local PT datasets storing in the CI local Metadata Repository
- Remote control of the application is realized through Remote Method Invocation (RMI).



Java Cryptography Architecture &
Java Cryptography Extensions

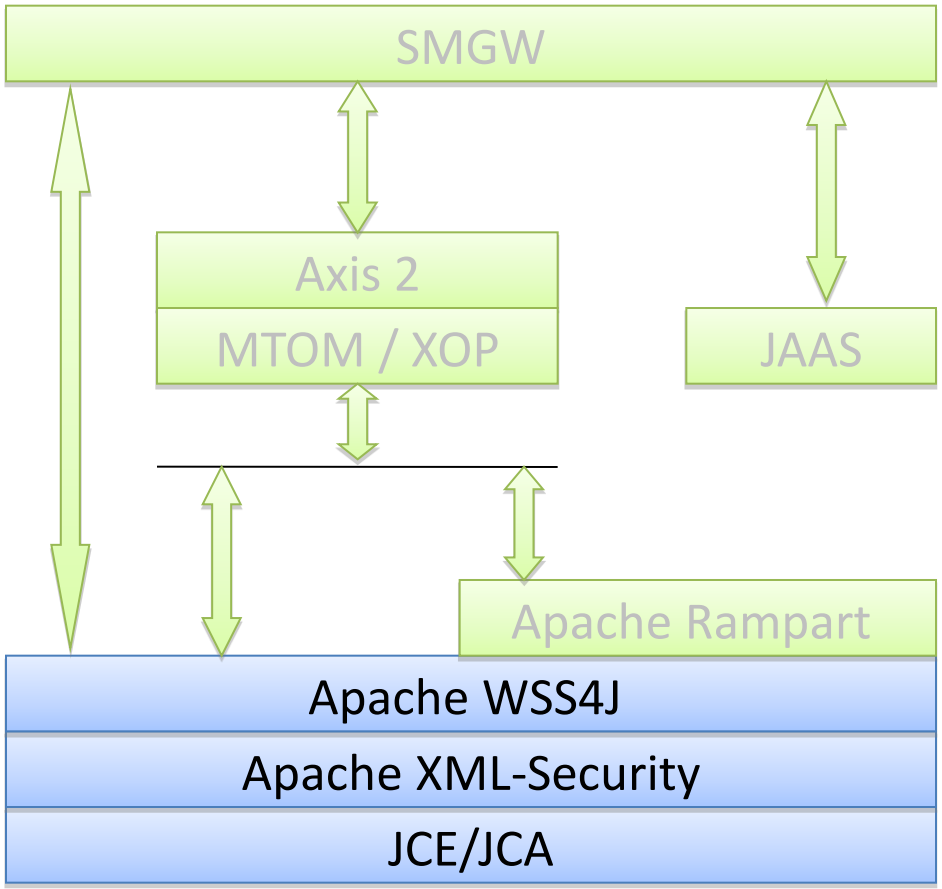
- *digital signature algorithms,*
- *message digest algorithms,*
- *key generation algorithms*
- *key factories,*
- *keystore creation & manag.*
- *algorithm parameter manag.*
- *algorithm parameter gen.*
- *certificate factories*



The Apache-XML-Security-J 1.4 supports

JSR-105: XML Digital Signature APIs for creating and validating XML Signatures.

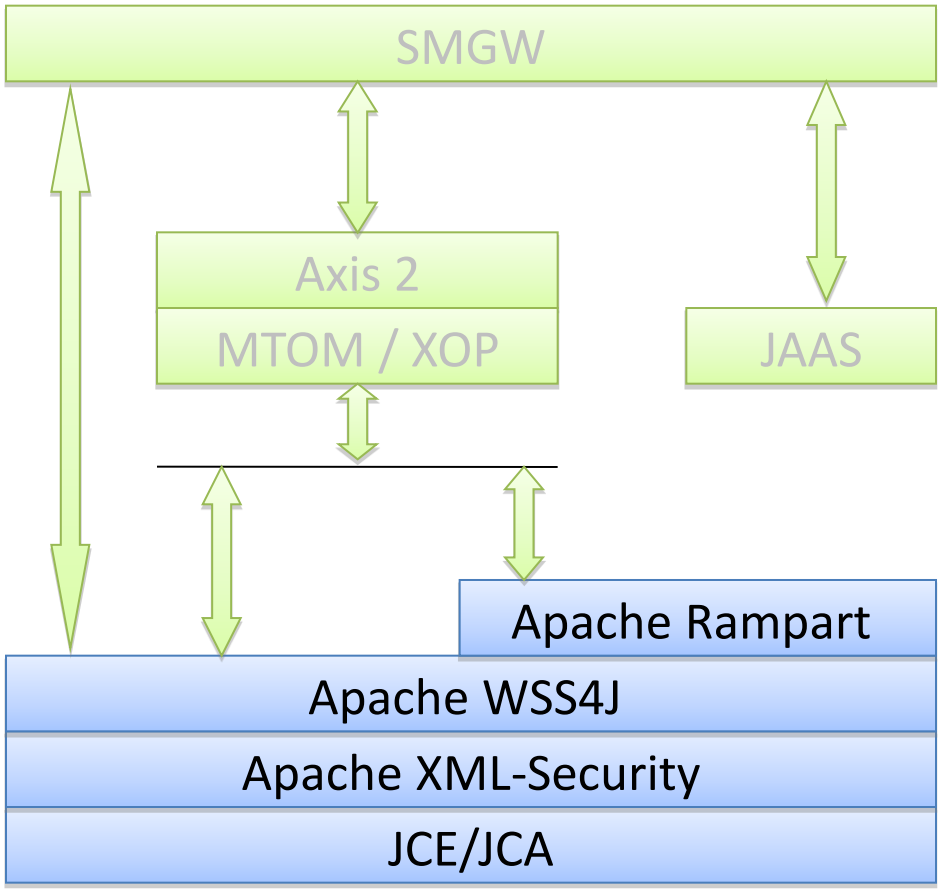
JSR-106: XML Digital Encryption APIs



WSS4J implements Web Services Security:

- *SOAP Message Security*
- *Username Token Profile*
- *X.509 Certificate Token Profile*

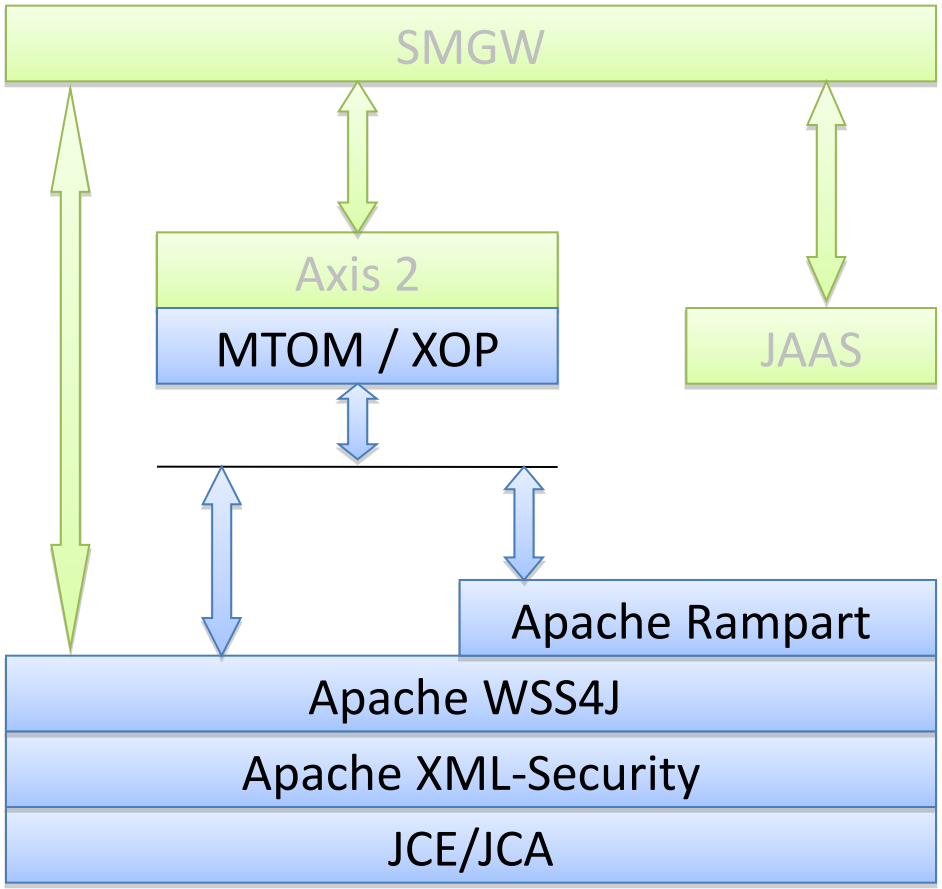
Security Technology Stack



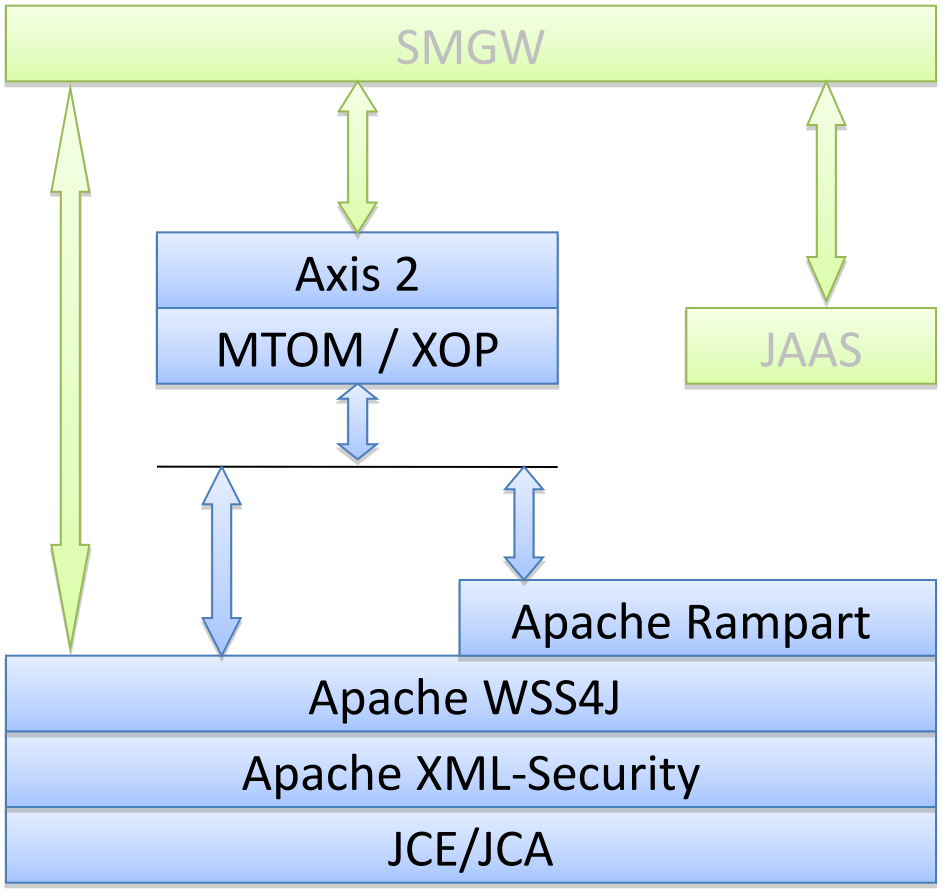
Rampart secures SOAP messages according to specifications in the WS-Security stack

Rampart supports

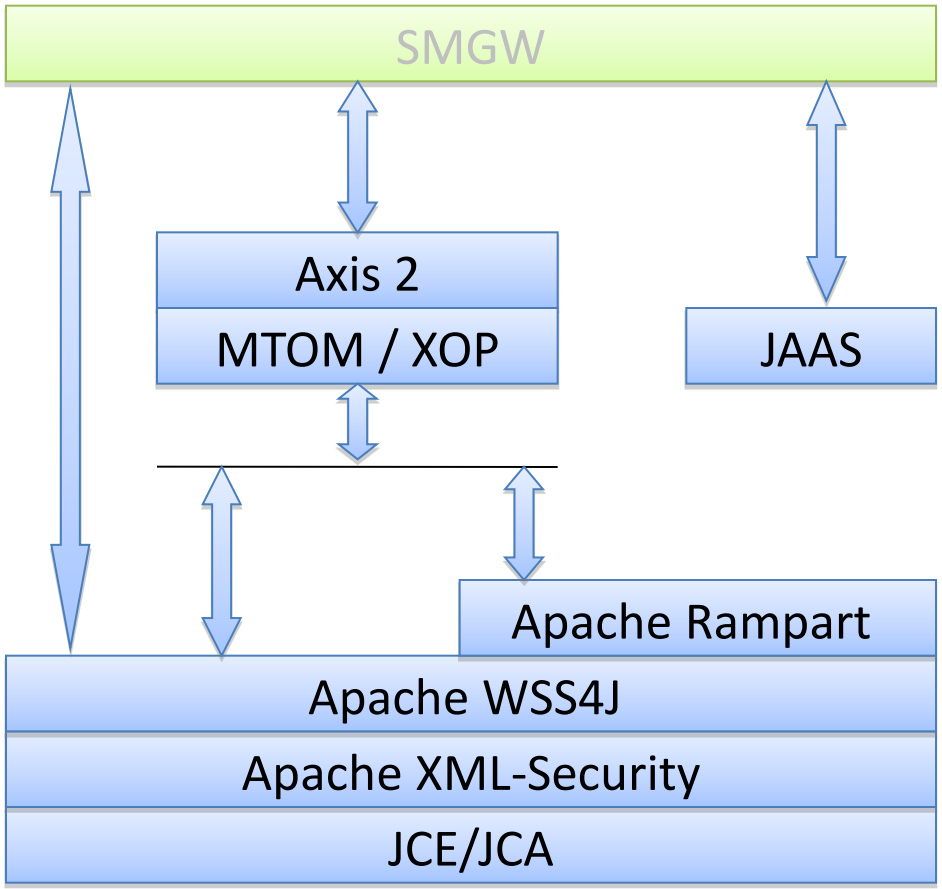
- *WS - Security*
- *WS - Security Policy*
- *WS - Trust*
- *WS-SXSAML*
- *SAML*



SOAP Message Transmission Optimization Mechanism optimizes the transmission and/or wire format of a SOAP message



Axis2 (Service Engine): provides the core web services engine and supports SOAP 1.1, SOAP 1.2, and REST-style services.
The Axis engine is designed as a series of loosely coupled message handlers, which perform pre- and post-processing on messages.

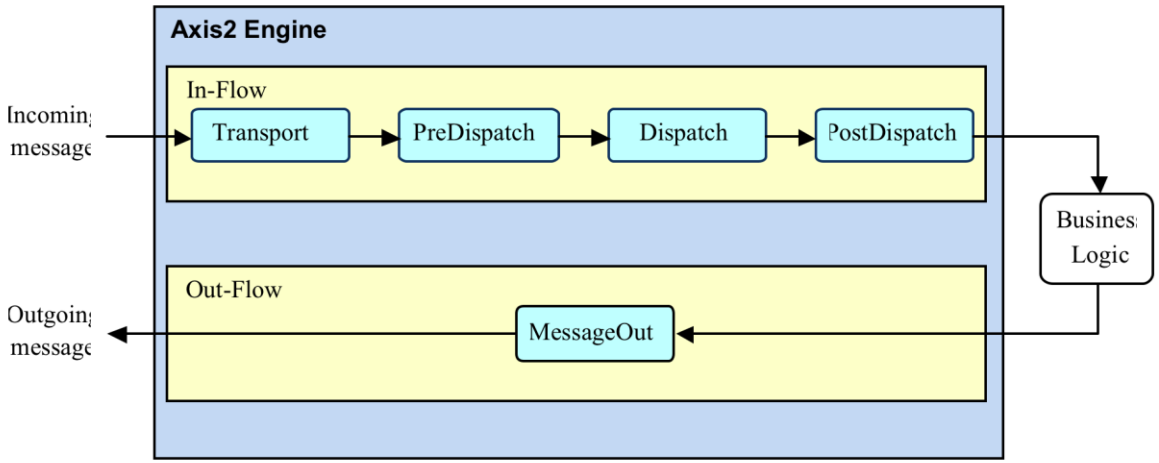


The Java Authentication and Authorization Service (JAAS) can be used for

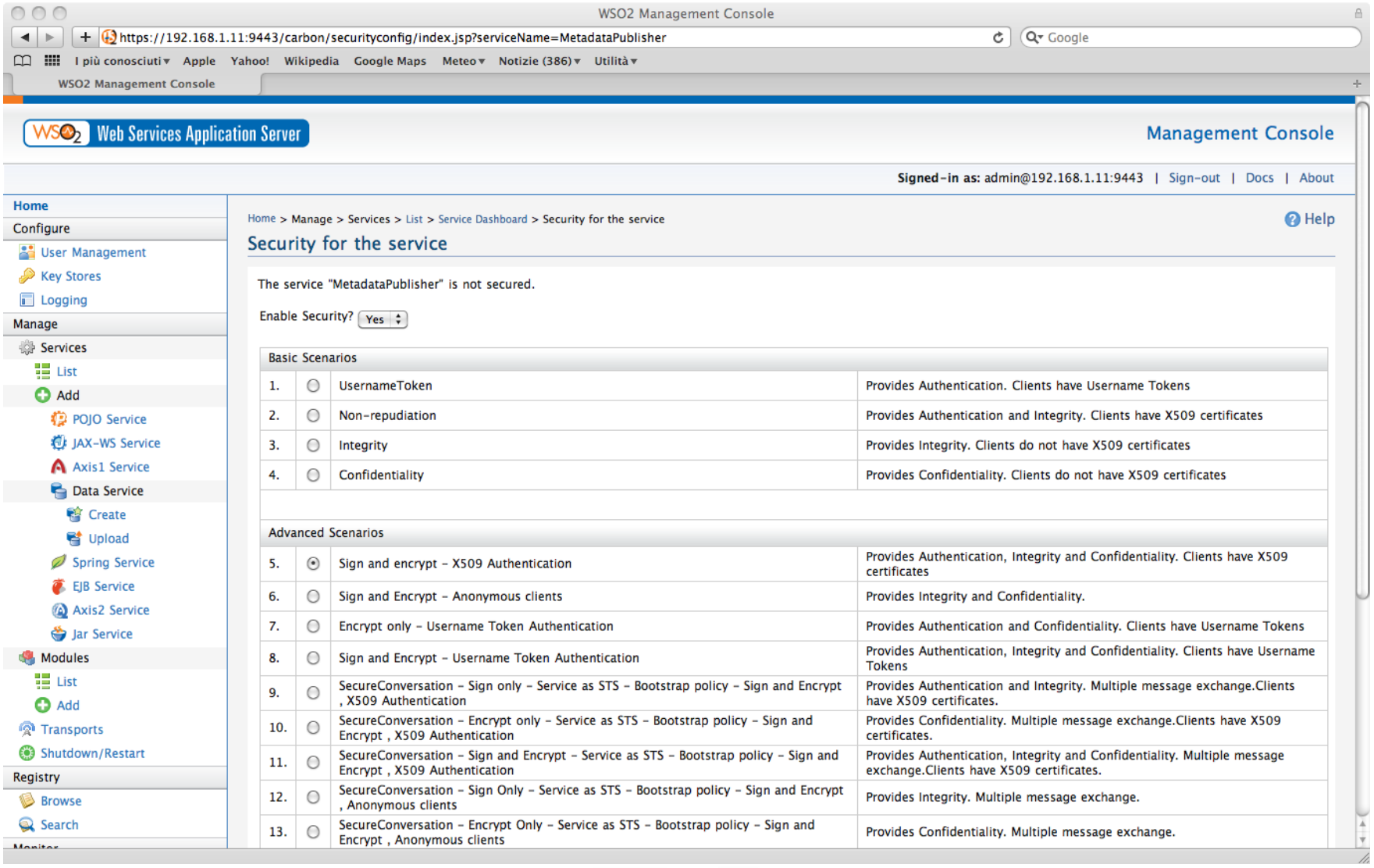
- *authentication of users, to reliably and securely determine who is currently executing Java code,*
- *authorization of users to ensure they have the access control rights (permissions) required to do the actions performed.*

Axis2 Service Engine

- The Axis2 service engine provides the core web services engine and supports SOAP 1.1, SOAP 1.2, and REST-style services.
- The Axis engine is designed as a series of loosely coupled message handlers, which perform pre- and post-processing on messages.
- In Axis2 the processing of incoming and outcoming messages is organized in a set of ordered phases named flow.
- Each phase is constituted by and ordered collection of handlers.
- A handler is the smallest unit of invocation in the Axis2 engine.

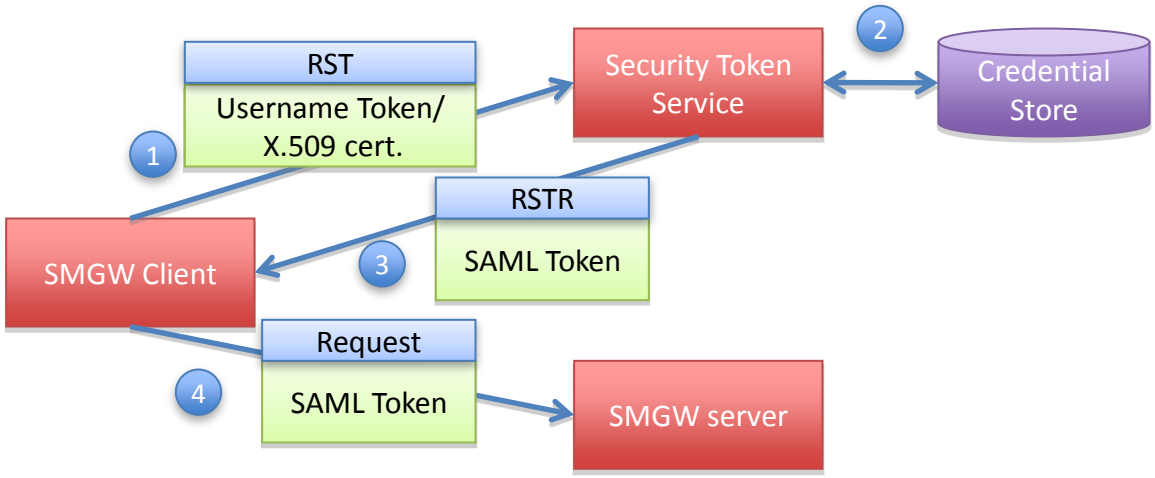


Service Security Management

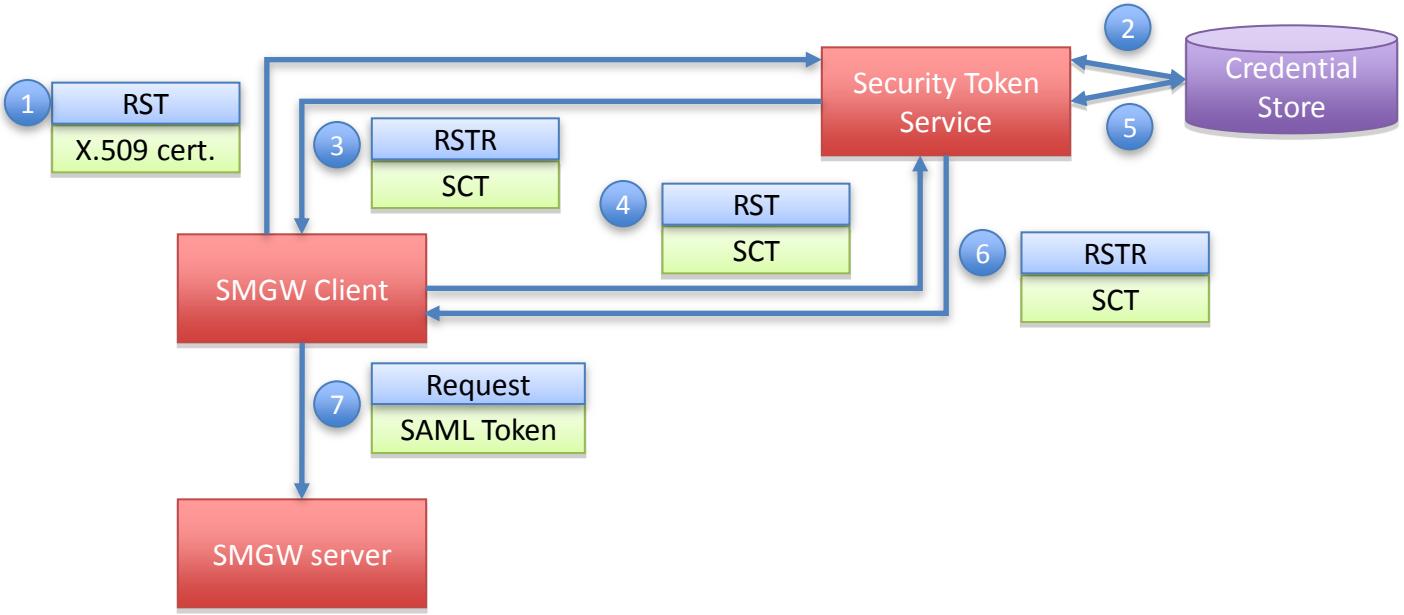


Security Assertion Markup Language

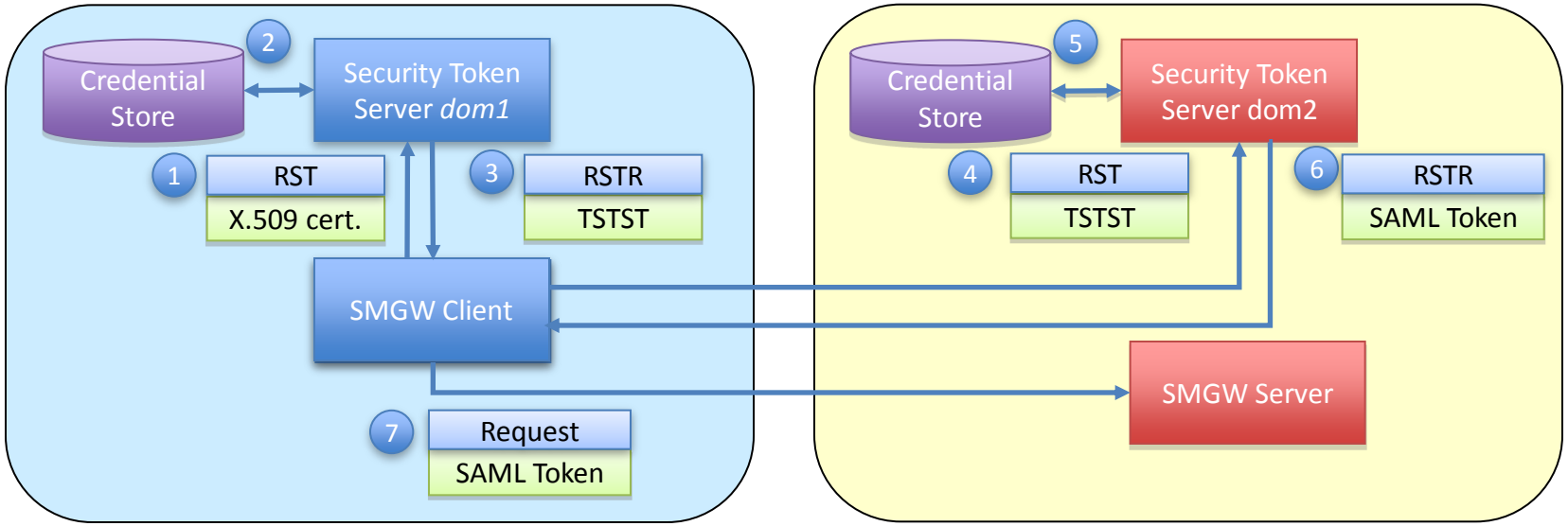
- SAML provides an XML-based framework for creating and exchanging security information between online partners.
- When a Security Token Service (STS) trusted by both the client and the Web service is employed, the client sends a Request Security Token (RST) message to the STS.
- After having verified the credentials, the STS sends back a Request Security Token Response (RSTR) message containing a security token asserting that the client has been authenticated.
- The security token is then employed by the client to gain access to the Web service.
- To authenticate the client, the Web service verifies that the token was issued by a trusted STS.



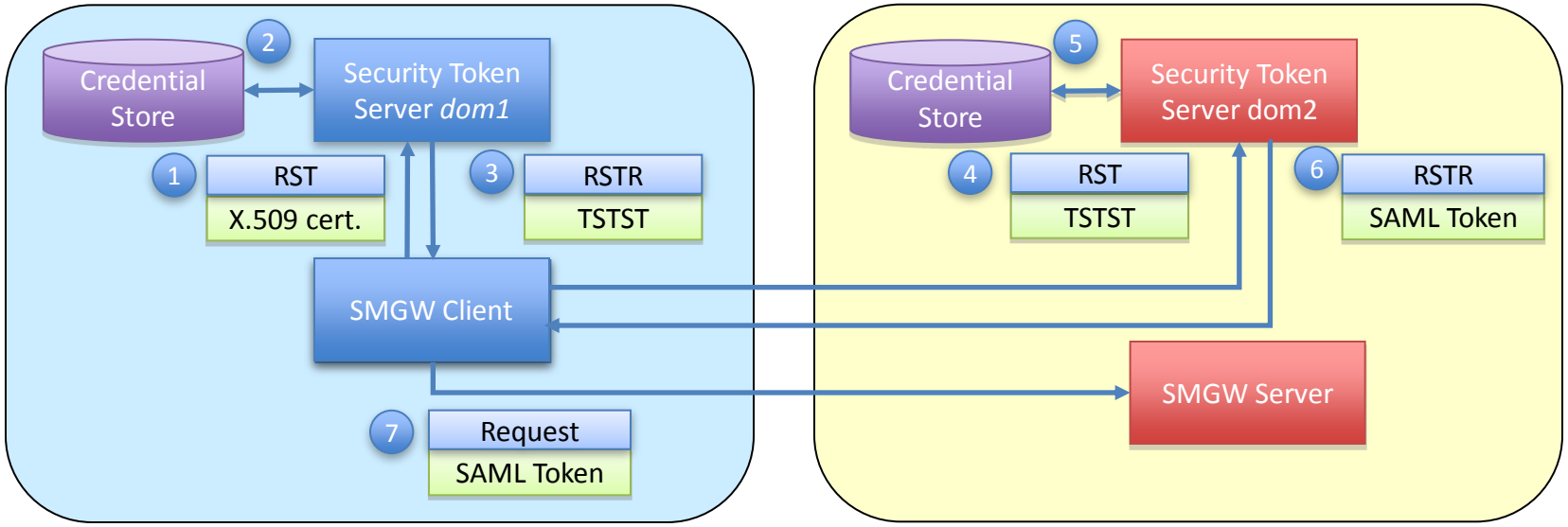
- Since an SMGW has to gain access to the subscribed SMGW Web services, each time the prediction tool performs a prediction update, to improve the performance, a WS-Secure Conversation with the STS can be established.
- In essence, an SMGW client will obtain from the STS a Security Context Token (SCT) demonstrating that it has been authenticated and will cache it.
- Then, the client will use the SCT to request a service token for gaining access to a service provided by a specific SMGW.



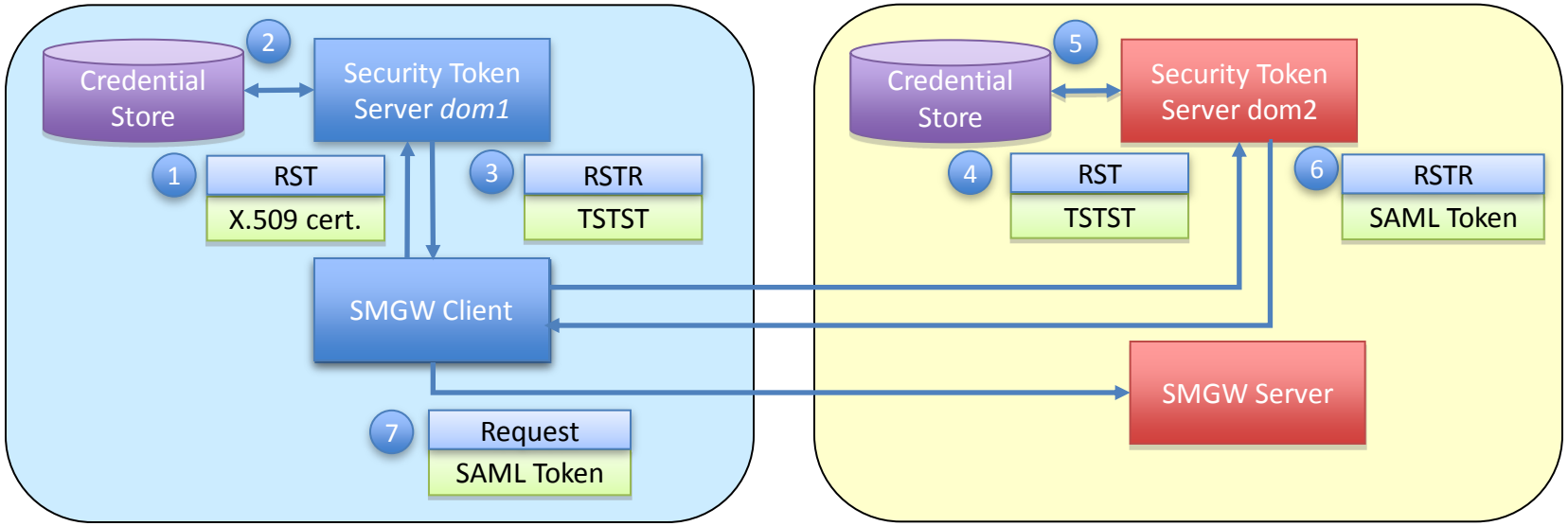
- An SMGW client could be authenticated by a security broker residing in its own security domain, and authorized and audited within the security domain where the serving SMGW operates.



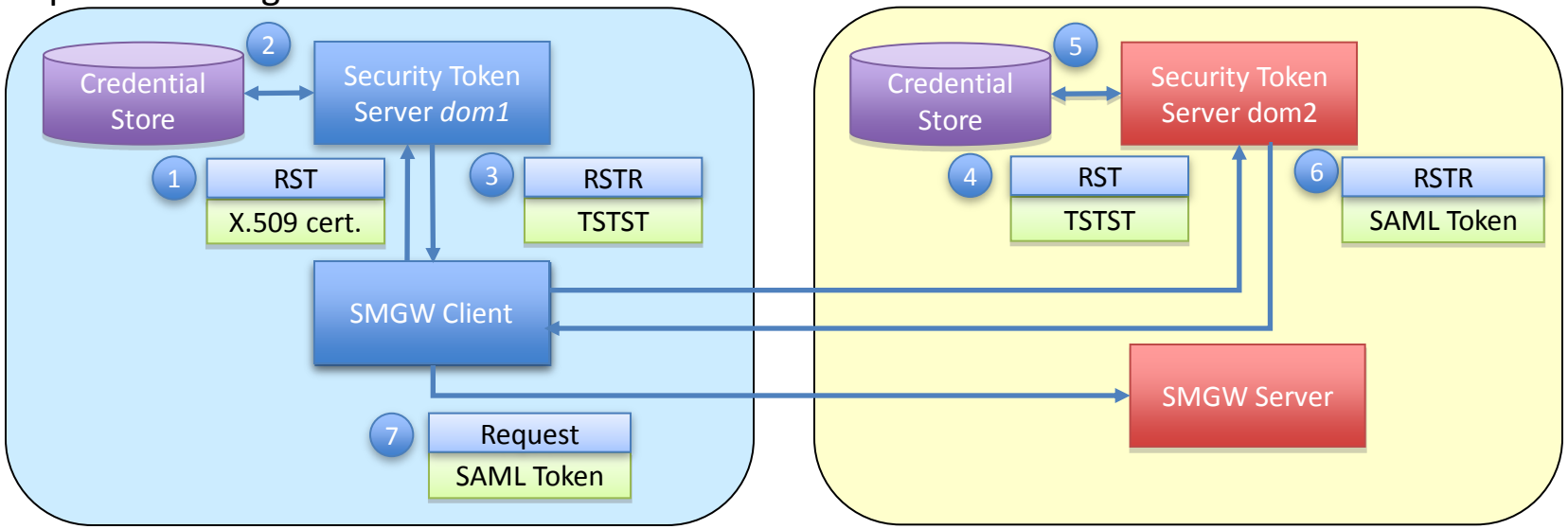
- The client issues a request to the STS residing in its own security domain for a security token to communicate with the STS in the SMGW server security domain.
- The STS in the client's domain authenticates the client and eventually issues a security token to the client for use in the security domain where the SMGW server resides.
- The client requests a security token from the STS in the SMGW server security domain by presenting the token issued by its STS.



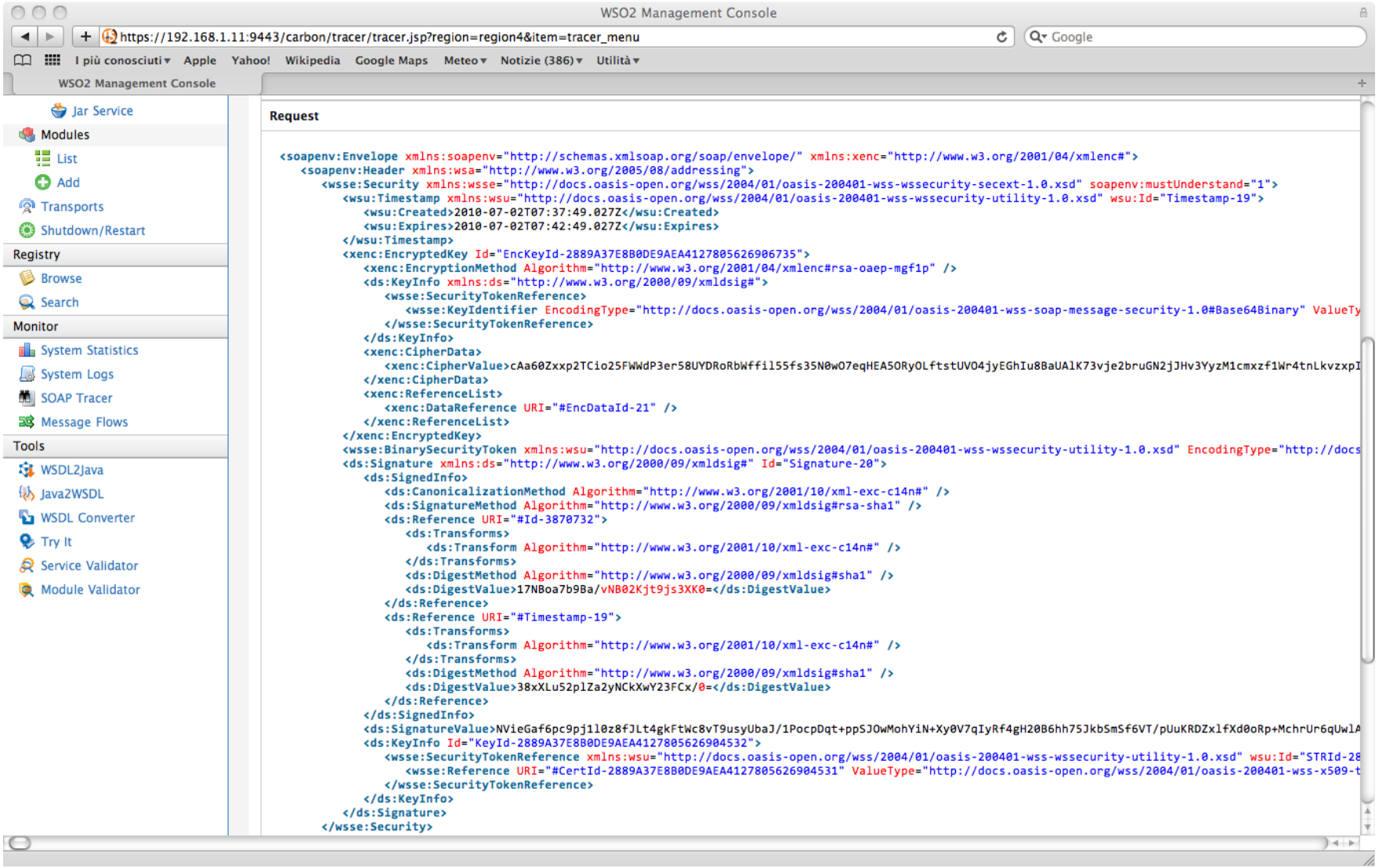
- The SMGW server STS validates the client's security token and then issues a security token that can be used to communicate with the service.
- The SMGW server STS validates the client's security token and then issues a Security Context Token (SCT) to the client. The SCT can be used by the client each time additional security tokens are required. The scope of the issued SCT is limited to the issuing STS regardless of whether the client specified the scope in the initial RST. And it can not be employed to directly gain access to a service.
- The client caches the SCT.



- The client requests a service token from the SMGW server to communicate with the service by means of the SCT.
- The SMGW server determines whether the client is authorized to access the service and eventually, issues to the client a security token that is used to authenticate with the service.
- The client attaches the security token issued by the SMGW server STS to the request and sends it to the SMGW server.
- The SMGW server validates the security token attached to the request and initializes and sends a response message to the client.



Soap Tracer


WSO2 Management Console
https://192.168.1.11:9443/carbon/tracer/tracer.jsp?region=region4&item=tracer_menu
Google
I più conosciuti
Apple
Yahoo!
Wikipedia
Google Maps
Meteo
Notizie (386)
Utilità
WSO2 Management Console
Jar Service
Modules
List
Add
Transports
Shutdown/Restart
Registry
Browse
Search
Monitor
System Statistics
System Logs
SOAP Tracer
Message Flows
Tools
WSDL2Java
Java2WSDL
WSDL Converter
Try It
Service Validator
Module Validator
Request
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">
<wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-19">
<wsu:Created>2010-07-02T07:37:49.027Z</wsu:Created>
<wsu:Expires>2010-07-02T07:42:49.027Z</wsu:Expires>
</wsu:Timestamp>
<xenc:EncryptedKey Id="EncKeyId-2889A37E8B0DE9AEA4127805626906735">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference>
<wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueTy
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>cAa60Zxxp2TCio25FWWdP3er58UYDRoRbWffil55fs35N0wO7eqHEA5ORyOLftstUVO4jyEGhIu8BaUAlK73vje2bruGN2jJHv3YyzM1cmxzf1Wr4tnLkvzxpI
</xenc:CipherData>
<xenc:ReferenceList>
<xenc:DataReference URI="#EncDataId-21" />
</xenc:ReferenceList>
</xenc:EncryptedKey>
<wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature-20">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
<ds:Reference URI="#Id-3870732">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>17NBoa7b9Ba/vNB02Kjt9js3XK0=</ds:DigestValue>
</ds:Reference>
<ds:Reference URI="#Timestamp-19">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
<ds:DigestValue>38xXLu52plZa2yNCkXwY23FCx/0=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>NVieGaf6pc9pj1l0z8fJLt4gkFtWc8vT9usyUbaJ/1PocpDqt+ppSJOwMohYiN+Xy0V7qIyRf4gH20B6hh75JkbSmSf6VT/pUuKRDZxlfXd0oRp+MchrUr6qUwlA
<ds:KeyInfo Id="KeyId-2889A37E8B0DE9AEA4127805626904532">
<wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="STRId-28
<wsse:Reference URI="#CertId-2889A37E8B0DE9AEA4127805626904531" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-t
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>

Conclusions

- OASIS Standards for WS security constitute viable means to implement SECURE MEDIATION in federated systems.
- To face the increase in the amount of exchanged data when full interdependence is considered, multicast mode should be implemented
- Securing multicast (XML) flows in federated systems is still an open issue.
- Assuring that an untrusted third party will not access sensitive information through interdependency analysis, a complex policy for controlling metadata exchange is required…

Questions?